REMARKS

Claims 1-21 remain pending in the case.

Objection to Drawings

In paragraph 3, the Office Action objected to the drawings. Applicants have submitted formal drawings herewith. Therefore, Applicants believe that the objection to the drawings has been overcome.

Objection to the Specification

In paragraph 4, the Office Action states "The disclosure is objected to because the following informalities: there appears to be a typographical error in page 11, line 7. It appears that '301' needs to be corrected to '305' as '301' indicates the instrument or." A replacement section has been provided herein correcting the typographical error. Therefore, Applicants believe that the objection to the specification has been overcome.

Double Patenting Rejection

Claims 1, 6, 7, 8, 13, 14, 15, 20, and 21

At paragraph 6 of the Office Action, Claims 1, 6, 7, 8, 13, 14, 15, 20, and 21 are rejected under the judicially created (non-statutory) doctrine of obviousness-type double patenting over Claims 1, 15, 21, 23, 29, and 31 of co-pending Application No. 10/016,955 (US Pub. No. 20030115582). A terminal disclaimer in compliance with 37 CFR 1.321 is being submitted concurrent with the instant response, thereby obviating the double patenting rejection.

102(b) Rejections

Claims 1-21

At paragraph 8 of the Office Action, Claims 1-21 are rejected under 35 U.S.C. 102(b) as being anticipated by Applicants admitted prior art. Applicants have reviewed the instant application and respectfully asserts that the claimed embodiments of the present invention are not anticipated by the prior art for the following rationale.

Claim 1 recites "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled." The prior art does not teach or suggest dynamic registration of first unwind information let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1. Since, embodiments of the present invention provide "...an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited in Claim 1, a target application does not need to be recompiled or relinked in any special way. For example, concerning "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," the instant application states on page 12, lines 5-10:

In prior art approaches, because this target application is not recompiled or relinked in any special way there was no direct support for the use of the pseudo-modules. The present embodiment, however, provides API 310 which now enables, for example, dynamic loader 308 to register the data within pseudo-module 302.

Therefore, Applicants respectfully submit that the prior art section of the instant application neither teaches nor suggests, among other things, "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1. Therefore, at least these reasons Applicants traverse the rejection under 35 USC 102(b) on the basis of the prior art section of the instant application.

At paragraph 9 of the Office Action, Claims 1-21 are rejected under 35 U.S.C. 102(b) as being anticipated by "HP Caliper-An Architecture for Performance Analysis Tool" by Hundt et al. (referred to hereinafter as "HP Caliper"). Applicants respectfully assert that the claimed embodiments of the present invention are neither taught nor suggested by HP Caliper for the following rationale.

Claim 1 recites "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled." HP Caliper does not teach or suggest registering of any kind let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1. Applicants have reviewed HP Caliper and respectfully submit that HP Caliper does not teach or suggest anything about unwind information or registering data let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1.

For example, in the abstract HP Caliper states "HP Caliper is an architecture for software developer tools that deal with executable (binary) programs HP Caliper teaches an architecture. It provides a common framework that allows building of a wide variety of tools for doing performance analysis, profiling, coverage analysis, correctness checking, and testing... This paper describes HP Caliper for HP-UX, running on the Intel® Itanium™ processor." Further, the first paragraph of the introduction states "The Intel Itanium processor instruction set architecture (ISA) offers an impressive set of architectural features which explicitly create synergy between compilers and the processor [10]." Note, HP Caliper does not mention anything about unwind information or registering data let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1. Therefore, HP Caliper does not teach or suggest "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1.

The Office Action states that HP Caliper discloses "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1 with the "Caliper API" at FIG. 1 and the section "4.1Algorithm" of HP Caliper. Although HP Caliper states that "HP Caliper API that consists of a set of C function interfaces" (refer to the first sentence of the second paragraph on page 3) that "interacts with a developer tool" (refer to Col. 1 second to last sentence of page 3), HP Caliper neither teaches nor suggests unwind information or registering data let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1. Therefore, for at least these reasons Applicants traverse the rejection under 35 USC 102(b) on the basis of HP Caliper.

At paragraph 10 of the Office Action, Claims 1-21 are rejected under 35 U.S.C. 102(b) as being anticipated by "Practicing JUDO: Java™ Under Dynamic Optimizations" by Cierniak et. al. (referred to hereinafter as "Cierniak"). Applicants respectfully assert that the claimed embodiments of the present invention are neither taught nor suggested by Cierniak for the following rationale.

Claim 1 recites "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled." Cierniak does not teach or suggest unwind information or registering data of any kind let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1. Applicants have reviewed Cierniak and respectfully submit that Cierniak does not teach or suggest anything about unwind information or registering data let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1.

For example, in the abstract Cierniak states "A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC)... In this paper, we present some static and dynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), I.e., lazy exceptions, lazy GC mapping, dynamic patching, and bounds checking elimination." Note, Cierniak does not mention unwind information or registering data let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1. Therefore, Applicants respectfully submit that Cierniak does not teach or suggest "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1.

The Office Action states that Cierniak discloses "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1 at page 19 section "5.1 Dynamic inline patching" and at page 20 "5.4 Lazy Exceptions." Note that the last sentence of the first paragraph of section "5.1 dynamic inline patching" on page 19 states, "Dynamic patching is a technique that patches the native code to preserve the correctness of the program once the assumptions made by the compiler are invalidated. The optimizing compiler produces the inlining code sequence as shown in Section 4.4.1. Right before code emission, the compiler replaces the cmp with a jmp, directly to the inlined code (as illustrated in Figure 5)." However, note that this section does not refer to, among other things, unwind information. Further, as previously stated in this reply with regards to "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled" the instant application states on page 12, lines 5-10:

> In prior art approaches, because this target application is not recompiled or relinked in any special way there was no direct support for the use of the pseudo-modules. The present embodiment, however, provides API

310 which now enables, for example, dynamic loader 308 to register the data within pseudo-module 302.

Therefore, Applicant respectfully submits that section "5.1 Dynamic inline patching" of Cierniak does not teach nor suggest "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1.

Further, note that the first sentence of the first paragraph of section "5.4 Lazy Exceptions" on page 20 states, "We have noted that some Java applications use exceptions to change the control flow only. In those applications the exception object's type is used to determine which handler catches the exception, but the content of the exception object is never accessed. In those cases creating the stack trade is unnecessary and we can save a substantial amount of work by never creating the exception object itself." Further in the second sentence of the first paragraph of Col. 2 on page 20 states "Our approach is to create exceptions lazily, i.e., initially assume that the exception object is not required...If the compiler can prove that the exception object is dead, the object is never created."

Although the instant application's title includes the word "lazy" and section "5.4 Lazy Exceptions" uses the word "lazy," section "5.4 Lazy Exceptions" does not teach or suggest unwind information or registering data let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by

Claim 1. Further, note that section "5.4 Lazy Exceptions" refers to a "...compiler can prove that the exception object is dead..." However, as previously stated in this reply, with regards to "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled" the instant application states on page 12, lines 5-10:

> In prior art approaches, because this target application is not recompiled or relinked in any special way there was no direct support for the use of the pseudo-modules. The present embodiment, however, provides API 310 which now enables, for example, dynamic loader 308 to register the data within pseudo-module 302.

Therefore, Applicant respectfully submits that neither section "5.1 Dynamic incline patching" nor section "5.4 Lazy Exceptions" of Cierniak teach nor suggest unwind information or registering data let alone "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled," as recited by Claim 1. For at least these reasons, Applicants traverse the rejection under 35 USC 102(b) on the basis of Cierniak.

Claims 8 and 15 should be allowable for similar reasons that Claim 1 should be allowable in that Claims 8 and 15 also recite "providing an application program interface which allows said data to be registered such that dynamic registration of said first unwind information and said first corresponding dynamically generated code is enabled."

Because Claims 2-7 depend from Claim 1, Claims 9-14 depend from Claim 8 , and Claims 16-21 depend from Claim 15 and contain additional limitations that are patentably distinguishable over the prior art section of the instant application, HP Caliper, and Ciernaik, these claims are also considered patentable over the prior art section of the instant application, HP Caliper, and Ciernaik. Therefore, Applicants respectfully submit that the basis for rejecting Claims 2-7, 9-14, 16-21 under 102(b) is traversed.

CONCLUSION

In light of the above remarks, Applicant respectfully requests reconsideration of the rejected Claims 1-21.

Based on the argument presented above, Applicant respectfully asserts that Claims 1 through 21 overcome the rejections of record and, therefore, allowance of these Claims is respectfully solicited.

The Examiner is invited to contact Applicants' undersigned representative if the Examiner believes such action would expedite resolution of the present Application.

Respectfully submitted,

WAGNER, MURABITO & HAO LLP

Date: 2/2/05

[illegible]

John P. Wagner Jr.
Reg. No. 35,398

Two North Market Street
Third Floor
San Jose, California 95113
(408) 938-9060